Naval Research Laboratory
Stennis Space Center, MS 39529-5004

NRL/FR/7441--96-9651

An Initial Design of Extended Vector Product Format for Modeling and Simulation

KEVIN SHAW

Mapping, Charting, and Geodesy Branch
Marine Geosciences Division

MAHDI ABDELGUERFI
EDGAR COOPER
CHRIS WYNNE

University of New Orleans
New Orleans, LA

H. VINCENT MILLER

Mississippi State University
Stennis Space Center, MS

BARBARA RAY
ROBERT BROOME
TOM FETTERER

Planning Systems Incorporated
Slidell, LA

August 26, 1996

REPORT DOCUMENTATION PAGE

Form Approved
OBM No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	August 26, 1996	Final

4. TITLE AND SUBTITLE

An Initial Design of Extended Vector Product Format for Modeling and Simulation

5. FUNDING NUMBERS

Job Order No. 574590806
Program Element No. RDT&EDA
Project No.
Task No.
Accession No. DN153251

6. AUTHOR(S)

Kevin Shaw, Mahdi Abdelguerfi*, Edgar Cooper*, Chris Wynne*, H. Vincent Miller†, Barbara Ray††, Robert Broome††, and Tom Fetterer††

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Research Laboratory
Marine Geosciences Division
Stennis Space Center, MS 39529-5004

8. PERFORMING ORGANIZATION REPORT NUMBER

NRL/FR/7441--96-9651

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Naval Research Laboratory
Marine Geosciences Division
Stennis Space Center, MS 39529-5004

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

*University of New Orleans, New Orleans, LA; †Mississippi State University, Stennis Space Center, MS; ††Planning Systems Incorporated, 115 Christian Lane, Slidell, LA

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The Digital Mapping, Charting, and Geodesy Analysis Program (DMAP) has been tasked with making the Defense Mapping Agency's (DMA) georelational Vector Product Format (VPF) more amenable to the Modeling and Simulation community. Under the direction of DMA's Terrain Modeling Program Office and the Defense Modeling and Simulation Office, DMAP has used requirements survey analyses and user feedback to devise a draft Extended VPF (EVPF), utilizing the design constraint of a relational database model. EVPF incorporates a new primitive structure for incorporating a Triangulated Irregular Network (TIN) as a means of representing three-dimensional surfaces. While VPF allows for the storage of three dimensions, this concept has traditionally been exploited primarily as elevation attributes of features. EVPF also stresses the use of attribute tables for storing rendering information. Using variable-length attribute fields, a current VPF capability not used to its fullest extent is also recommended for richer feature attribution.

14. SUBJECT TERMS

digital MC&G, performance analysis, modeling and simulation

15. NUMBER OF PAGES

24

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	Same as report

CONTENTS

EXECUTIVE SUMMARY

With support from the Defense Mapping Agency's (DMA) Terrain Modeling Program Office and the Defense Modeling and Simulation Office, the Digital Mapping, Charting, and Geodesy Analysis Program (DMAP) has investigated modifications to DMA's current Vector Product Format (VPF) that would benefit the Modeling and Simulation (M&S) community while preserving the relational database model approach of VPF, a design constraint. In its present form, VPF has been documented as not meeting requirements of this particular community, mostly in its treatment of three-dimensional (3-D) data. DMAP proposes an extended VPF (EVPF), whereby 3-D surfaces may be stored in VPF's georelational format. As pointed out by DMAP, the VPF standard currently possesses the capability of storing Triangulated Irregular Networks (TINs), a popular format for representing 3-D surfaces. However, after an investigation of this VPF-based structure, as well as triangle-based and vertex-based data structures, DMAP has concluded that an improved triangle-based data structure for incorporating TINs is optimal with respect to access time, coupled with only a marginal increase in storage space when compared to the standard triangle-based data structure. The use of primitive attribute tables and multivalued attributes is also proposed. These extensions should provide the M&S community with a basic format for utilizing vector data.

AN INITIAL DESIGN OF EXTENDED VECTOR PRODUCT FORMAT FOR MODELING AND SIMULATION

1.0 INTRODUCTION

The Extended Vector Product Format (EVPF) (Shaw et al. 1996) is a georelational database format designed specifically to meet the vector requirements of the Modeling and Simulation (M&S) community. Its most prominent modification to the Vector Product Format (VPF), its precursor, is the addition of three-dimensional (3-D) representation of surfaces of objects. This representation is based on a new set of tables defined to support a Triangulated Irregular Network (TIN). With this new representation, a surface (e.g., a terrain) can be modeled with relational tables similar to those offered by VPF.

Other extensions include a method of introducing attribution which, by its nature, is useful only in the visual representation of objects. These are similar to what VPF defines as the feature attribute table. The primitive attribute table allows for the storage (directly, or indirectly via a related attribute table) of information useful in the rendering of objects.

Finally, to fully describe features, EVPF allows for specific attributes to be multivalued. Variable-length fields are used to model such situations.

2.0 TRIANGULATED IRREGULAR NETWORKS

Rectangular grids and TINs are the two primary techniques used to represent digital elevation. Rectangular grids, such as those used in the Defense Mapping Agency's (DMA) (Littlefield 1995) Digital Terrain Elevation Data (DTED) databases, are characterized by their simplicity and regularity. Because rectangular grids are bound by their regularity, they do not adapt to the complexity of the terrain being modeled.

The TIN model approximates a topographic surface using a network of planar, nonoverlapping, and irregularly shaped triangle faces (Floriani 1987). The irregular shape of the triangles allows TINs to easily adapt to the roughness of the terrain, thus providing a surface representation using a limited amount of data. For instance (Polis and McKeown 1992), a DTED rectangular grid composed of 90,000 nodes is reduced to only 563 nodes using TINs. The ability of the TIN model to adjust its resolution based on the complexity of the terrain being modeled makes it more efficient in a wide range of applications, including real-time display and automated terrain analysis (Polis and McKeown 1992; Polis and McKeown 1995). Another important advantage of the TIN model is that it can incorporate surface-specific constraints such as prespecified linear and area features (Floriani 1987; Floriani 1989).

One goal of this study is to extend the VPF (Department of Defense 1993) to allow for the efficient storage and access of TIN-based elevation data. It is a further goal of this study to investigate the efficient integration of terrain elevation data with ground surface features. Finally, EVPF is to provide a fully 3-D data structure for improved modeling and simulation exploitation.

Using VPF's connected-node, edge, and face geometric primitives, TINs can be encoded. However, the Digital Mapping, Charting, and Geodesy Analysis Program (DMAP) will show in this report that such a scheme is inefficient. Two alternate encoding schemes, the vertex-based and triangle-based data structures, are evaluated. The evaluation takes into account the storage requirement of each data structure and the time complexity to retrieve those basic relationships (between primitives) that are not explicitly encoded into the data structure. An improved triangle-based data structure is shown to exhibit better overall performance.

This report's analysis of TINs is organized as follows. In Secs. 3.0 and 4.0, the VPF's winged-edge topology, the vertex-based, triangle-based, and improved triangle-based data structures are introduced and compared based on their ability to efficiently encode TIN-based elevation data. An extended VPF (EVPF) is introduced in Sec. 5.0. Handling TIN elevation data in the presence of tiles is presented in Sec. 6.0. A comprehensive example that includes TIN elevation data and linear and area features is given in Sec. 7.0.

3.0 DATA STRUCTURE SCHEMATAS FOR TINS

Triangles, edges, and vertices (also called nodes) are the three primitives of the TIN topographic surface model. Triangles and edges are similar to VPF's faces and edges (Department of Defense 1993) except that:

- A TIN's triangle is bounded by exactly three edges, three nodes, and three adjacent triangles;
- Unlike an arbitrary face, a TIN's triangle cannot have an edge inside;
- Each triangle's edge is completely characterized by its two extreme nodes.

Clearly, an efficient data structure to encode TINs should take advantage of these three characteristics.

As stated in (Floriani 1989 and Woo 1985), nine adjacency relations can be defined between pairs of primitives (Fig. 1). In Fig. 1, a directed arc represents an ordered relation between two primitives. For example, the relation **edge** → **node** stores the two extreme nodes of each edge. A data structure to encode TINs will, in general, combine the three primitives with a subset of the adjacency relations. Clearly, encoding more adjacency relations leads to an increase in the amount of storage requirement but improves the data access efficiency. Altogether, there are more than five hundred possible schematas forming eight storage classes (Woo 1985). Some adjacency relations,

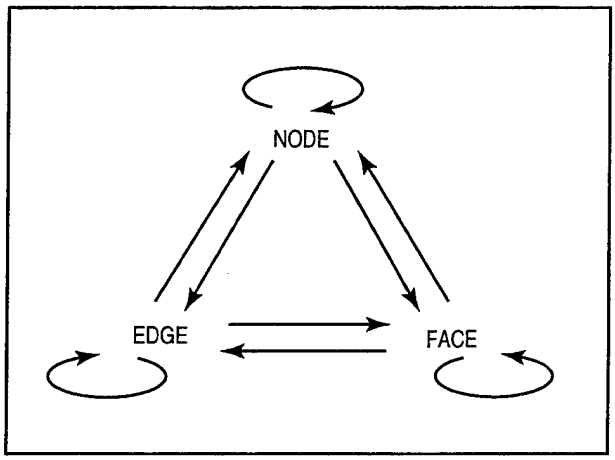

Fig. 1 — Nine relations between three geometric primitives

such as **node → edge**, are one-to-many and, as a result, are less convenient to store than constant relations. This has lead to the introduction of partial (also referred to as fractional) relations (Baumgart 1972). For instance, the relation **face → edge** can be partially stored by having a face point to only one of its many possible edges. This partial relationship between a face and its enclosing edges is used in VPF's winged-edge topology (Department of Defense 1993). It is noted that the introduction of partial relationships between the three primitives leads to a dramatic increase in the number of possible schematas.

The most common data structures to encode TINs are the vertex-based and triangle-based structures (Floriani 1989; Jones et al. 1994). VPF's winged-edge topology can also be utilized to store TINs. In the following sections, the performance of these three data structures will be investigated. The performance of a TIN data structure is evaluated in terms of its storage requirement and the time complexity to retrieve those basic relationships (between geometric primitives) that are not directly encoded into the data structure. Woo (Woo 1985) identifies nine access primitives (Table 1). Some of these access primitives form the basis of geometric algorithms used to perform tasks such as contour extraction and interpolation.

3.1 Encoding TINs Using VPF's Winged-Edge Topology

In VPF, an arbitrary face is stored using the structure shown in Fig. 2. These relationships include an edge table that stores:

- A unique edge ID;
- The one-to-many relationship between an edge and its nodes;
- A partial relationship between an edge and the edges connected to it; and
- The relationship between an edge and its two adjoining faces, referred to in VPF as left face and right face.

In VPF's level 3 topology (Department of Defense 1993), each edge stores pointers to two of the edges to which it connects. These two edges, known as left edge and right edge, are determined by the edge direction as shown in Fig. 3.

Table 1 — Nine Access Primitives

ACCESS PRIMITIVE	DESCRIPTION
AC1	Given triangle i find its three nodes
AC2	Given triangle i find its three edges
AC3	Given triangle i find its three adjacent triangles
AC4	Given node i find the t_i triangles in which it's contained
AC5	Given node i find the n_i nodes to which it connects
AC6	Given node i find the e_i edges in which it's contained
AC7	Given edge i find its two extreme nodes
AC8	Given edge i find the e_i edges to which it connects
AC9	Given edge i find the two adjoining faces

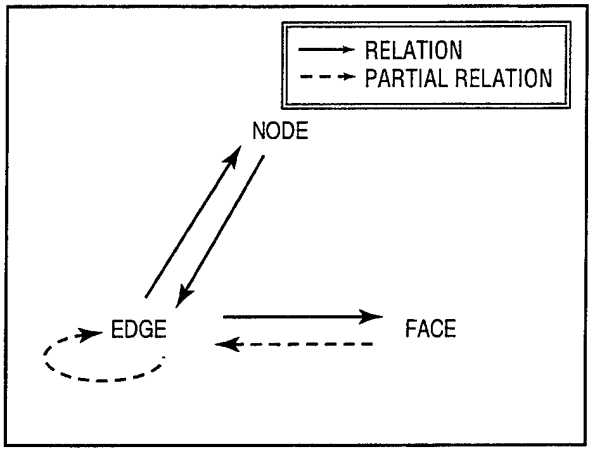

Fig. 2 — VPF's winged-edge topology relationships

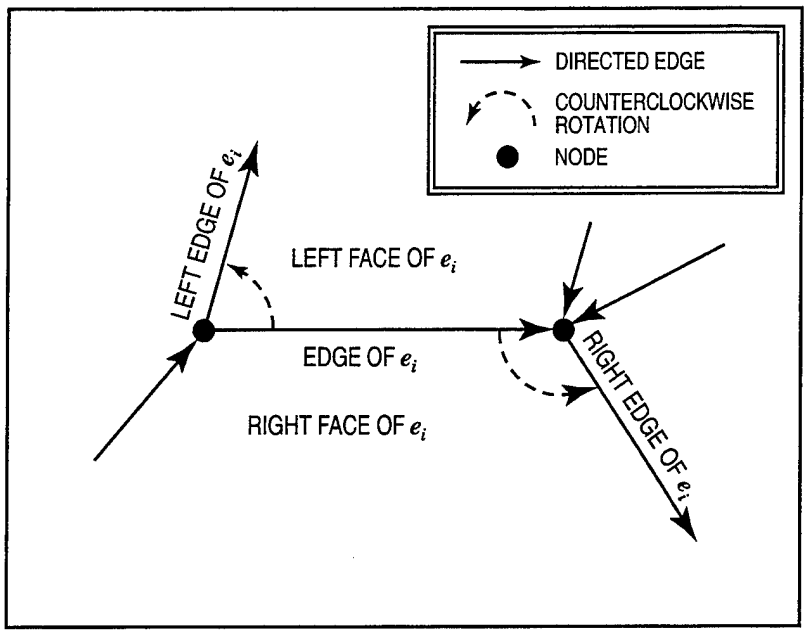

Fig. 3 — Attributes of an edge in VPF's winged-edge topology (level 3)

The node (referred to as a connected node in VPF) table stores the node's ID along with the node's coordinates. An additional column is used to specify a partial relationship between a node and the edges around it. This is achieved by associating with each node a pointer to one of its containing edges (called the first edge). This relationship allows the navigation around a node, and as a result, makes the implementation of access primitives AC4, AC5, and AC6 more efficient.

The partial relationship between a face and one of its enclosing edges is implemented using two tables: a face table and a ring table. The use of these two tables allows the navigation around the face's enclosing edges (known as an outer ring in VPF) and around any inner ring embedded in the face. The face table stores

- A unique face ID number; and
- A pointer to the outer ring in the ring table.

The ring table points to one of the edges of each ring of a face. A ring table is composed of the following three columns:

- a ring ID number,
- a pointer to a face, and
- a pointer to one of the ring's edges (called the start edge).

In a ring table, the ordering of the rows is important. The first row in a ring table with a new face ID is always the face's outer ring. Any subsequent row with the same face ID is an inner ring.

These four VPF tables can be utilized without modification to store TINs. The sample triangular network of Fig. 4 is used for illustration. Partially constructed node, edge, face, and ring tables are shown in Tables 2, 3, 4, and 5, respectively.

3.2 Encoding TINs Using a Vertex-Based Data Structure

The vertex-based data structure stores the one-to-many relation **node → node**. Therefore, a variable number of fields needs to be allocated in the data structure. A partial encoding of the example of Fig. 4 is shown in Table 6. The vertex-based data structure stores a node ID, the node's three coordinates, and pointer to the neighboring nodes in clockwise (or counterclockwise) order.

Table 2 — Partial Connected Node Table for Fig. 4

ID	X	Y	Z	FIRST_EDGE
1	x_1	y_1	z_1	6
2	x_2	y_2	z_2	5
. . .	. . .	. . .	. . .	. . .

Table 3 — Partial Edge Table for Fig. 4

ID	START_ NODE	END_ NODE	RIGHT_ EDGE	LEFT_ EDGE	RIGHT_ FACE	LEFT_ FACE	COORDINATES
1	4	10	6	3	5	4	$x_4y_4z_4x_{10}y_{10}z_{10}$
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .

Table 4 — Partial Face Table for Fig. 4

ID	RING_POINTER
. . .	. . .
4	9
. . .	. . .

Table 5 — Partial Ring Table for Fig. 4

ID	FACE	START_EDGE
. . .	. . .	. . .
9	4	1
. . .	. . .	. . .

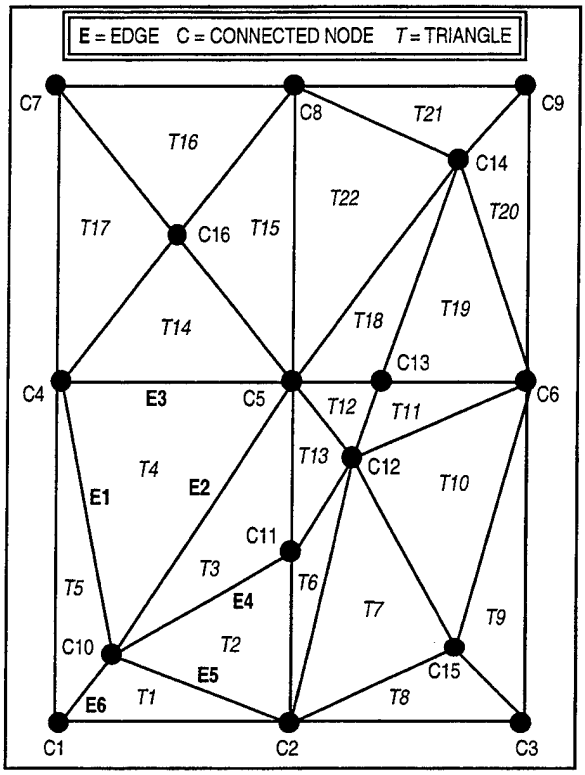

Fig. 4 — Sample TIN

Table 6 — Partial Encoding of the TIN Table Corresponding to Fig. 4 (Using a Vertex-Based Data Structure)

ID	X	Y	Z	VERTEX NEIGHBORS
1	x_1	y_1	z_1	2 10 4
2	x_2	y_2	z_2	3 15 12 11 10 1
. . .	. . .	. . .	. . .	. . .

The relation **node → edge** is implicitly stored in the table. As stated by Floriani (1989), this data structure completely characterizes a triangular subdivision without any ambiguity.

3.3 Encoding TINs Using a Triangle-Based Data Structure

In general, a triangle-based data structure encodes TINs using two tables: a node table and a TIN table. The TIN table stores the triangle ID number, pointers to the triangle's three vertices and pointers to the neighboring triangles. The pointers to the nodes and neighboring triangles are stored in consistent clockwise (or counterclockwise) order. As a result, the TIN-table explicitly stores, in a single table, the relations **face → node** and **face → face**. Table 7 shows the TIN table corresponding to Fig. 4 using a triangle-based data structure. Two other relationships, **node → edge** and **edge → face**, are implicitly stored in the TIN table.

3.4 Encoding TINs Using an Improved Triangle-Based Data Structure

As will be demonstrated in the next section, the triangle-based data structure implements the access primitives AC1, AC2, and AC3 optimally. However, the access primitives AC4, AC5, and AC6 cannot be implemented efficiently with this data structure. This is due to the fact that, with the triangle-based data structure, navigation around a given node in an efficient manner is impossible.

Table 7 — Partial Encoding of the TIN Table Corresponding to Fig. 4 (Using a Triangle-Based Data Structure)

ID	VERTEX 1	VERTEX 2	VERTEX 3	TRIANGLE 1	TRIANGLE 2	TRIANGLE 3
1	1	2	10	Null	2	5
2	2	11	10	6	3	1
. . .	. . .	. . .	. . .	. . .	. . .	. . .

To remedy this situation, DMAP proposes to modify the triangle-based approach in such a way that the resulting data structure implements primitives AC4, AC5, and AC6 in an efficient manner. The new structure will be referred to as an improved triangle-based data structure. The new data structure leaves the TIN table unchanged. However, the node table is augmented with a partial relationship between a node and its containing triangles. This is achieved by adding an extra field—referred to as first-TIN—in the node table. This first-TIN field will contain a pointer to one of the triangles containing the given node. This will permit the navigation around a node that results in an efficient implementation of AC4, AC5, and AC6. A partial node table of an improved triangle-based encoding of the sample TIN in Fig. 4 is shown in Table 8.

4.0 COMPARATIVE ANALYSIS OF THE TIN SCHEMATAS

In this section, the data structures introduced in the previous sections are evaluated in terms of their storage requirement and the time complexity to retrieve basic access primitives. Not all access primitives listed in Table 1 are applicable. For instance, the triangle-based data structure does not store edges explicitly and as a result access primitive AC7 is meaningless in this context. However, in the same context, AC7 and AC8 can be implemented efficiently if an edge is identified using its two extreme nodes. Without loss of generality, our time evaluation will be restricted to the first six basic access primitives.

The time performance of the four data structures is shown in Table 9. The time performance investigation follows the methodology outlined in (Woo 1985). When direct access to a relationship between geometric primitives is possible, the time access is said to be constant. Direct access means that the access to the relationship is performed by accessing an existing table using its primary key. For instance, in the triangle-based data structure, given a triangle ID i, it is possible to determine its three vertices in constant time. Indeed, using the TIN table's primary key i one could easily determine the three nodes around the triangle (access primitive AC1), especially if a spatial index were available. Constant access time constitutes a lower bound and will be referred to throughout as k. Alternatively, it may be necessary to access a relationship using a foreign key. In this case, the access is said to be a linear function of the number of nodes. For example, suppose that, in the case of the triangle-based data structure, we would like to determine the n_i nodes around node i (access primitive

Table 8 — Partial Encoding of the Connected Node Table Corresponding to Fig. 4 (Using an Improved Triangle-Based Data Structure)

ID	X	Y	Z	FIRST_TIN
1	x_1	y_1	z_1	1
2	x_2	y_2	z_2	2
. . .	. . .	. . .	. . .	. . .

Table 9 — Time Performance of Data Structures

(k = constant, n_i = number of nodes connected to node i, n = total number of TIN nodes, n/a = not applicable)

	AC1	AC2	AC3	AC4	AC5	AC6	TOTAL
VPF's Winged-Edge Topology	k	k	k	$O(n_i)$	$O(n_i)$	$O(n_i)$	$3(k + O(n_i))$
Vertex-Based	n/a	n/a	n/a	n/a	k	n/a	n/a
Triangle-Based	k	k	k	$O(n)$	$O(n)$	$O(n)$	$3(k + O(n))$
Improved Triangle-Based	k	k	k	$O(n_i)$	$O(n_i)$	$O(n_i)$	$3(k + O(n_i))$

AC5). Clearly this information can only be extracted from the TIN table. However, we are given only i, a foreign key in the TIN table. The only way to obtain the n_i nodes is by sequentially scanning the TIN table multiple times. In this case, the access time is a linear function of the number of TIN nodes n. Linear access time constitutes the worst-case (upper bound) access time and will be referred to as $O(n)$. Constant and linear time access are the two extreme cases. The use of partial relationships between arbitrary primitives yields an intermediate access time. Consider AC4 in the context of the improved triangle-based data structure. Here, we would like to determine the t_i triangles containing node i. Clearly, without the partial relationship between a node and the triangle primitives, the access time is $O(n)$. However, using this existing partial relationship, we can use the node ID to obtain, from the connected node table, the ID of one of the triangles around the given node. We can then use this triangle ID to obtain, from the TIN table, other triangles and nodes of interest. This process is repeated until all of the triangles around node i are obtained. The access time, in this case, is $O(n_i)$. In essence, the existence of this partial relationship allows for the navigation around a given node and, as a result, allows for the determination of the t_i triangles, e_i edges and n_i nodes around node i in $O(n_i)$ access time. Note that, in general, the number of TIN nodes n is very large compared to the n_i nodes connected to a given node i.

From Table 9, it is also seen that the triangle-based data structure is optimal with respect to AC1, AC2, and AC3. However, it has a worst-case performance when it comes to primitives AC4, AC5, and AC6. The total access time (cumulative access time of the six access primitives) $3(k + O(n))$ of the triangle-based approach does not compare favorably with the total access time $3(k + O(n_i))$ of VPF's winged-edge topology. However, notice that the total access time $3(k + O(n_i))$ of the improved triangle-based data structure represents a dramatic improvement from that of the traditional triangle-based data structure. This dramatic improvement is obtained with minimal additional storage cost (approximately an additional 0.5% storage as shown in the following paragraph).

The storage requirement of each data structure is presented in Table 10 and is expressed in terms of the number of fields required by each data structure. The figures shown in Table 2 are based on the fact that any planar triangulation of a set of n nodes, B of which belong to the convex hull (boundary), has exactly $3n - B - 3$ edges and $2n - B - 2$ triangles (Shamos 1978). It is noted that, in general, B is very small compared to n. VPF's winged-edge topology requires the largest amount of storage. It exceeds the storage requirement of the vertex-based and triangle-based data structures by a factor of about 6 and 3, respectively. To reduce the storage requirement of VPF's winged-edge topology, one could take advantage of the fact that, unlike an arbitrary face, a triangle cannot have an inside edge. Since, in this case, there are no inner rings, one could eliminate the ring table and replace (in the face table) the pointer to the ring table by a pointer to the face's

Table 10 — Storage Requirements for the Data Structures

(n = number of nodes in the TIN, B = number of nodes in the TIN's convex hull)

DATA STRUCTURE	STORAGE REQUIREMENT
VPF's Winged-Edge Topology	$54n - 18B - 49$
Vertex-Based	$10n - 2B - 6$
Triangle-Based	$18n - 7B - 14$
Improved Triangle-Based	$19n - 7B - 14$

starting edge. This will, however, result in only a modest 10% reduction in the storage requirement of VPF's winged-edge topology. The vertex-based data structure requires the least amount of storage. Note that the storage requirement of the triangle-based and improved triangle-based data structures are essentially the same (the difference is 0.5% in favor of the traditional triangle-based data structure). However, the time performance of the improved triangle-based data structure is, as shown previously, far superior to that of the traditional triangle-based data structure.

Since the vertex-based data structure stores a node and its surrounding nodes, it performs AC5 optimally. The main problem with the vertex-based data structure is that it is of little practical use in applications where area features are to be associated with triangles. This is because the vertex-based data structure does not store triangles. In contrast, since the triangle-based data structure stores (in the TIN table) a triangle, its three nodes, and its three edges (each pair of nodes implicitly represents an edge), features associated with a triangle, its nodes, and its edges can all be specified in a single feature table. For instance, a distributed rainfall-runoff model (Tachikawa et al. 1994), is encoded by using TINs and a triangle-based data structure. Several edge features such as side-index and side-component are stored. The side component-index specifies whether an edge belongs to a valley, slope, ridge, channel, or a boundary, and a side-attribute index specifies whether water flows out of an edge, along an edge, or into an edge.

The use of VPF's winged-edge topology has the advantage that it essentially does not yield any changes to VPF. However, this approach suffers from high storage cost. Additionally, this approach will require that the TIN triangles be stored along with arbitrary faces. This option is clearly not desirable.

The improved triangle-based data structure is as fast as VPF's winged-edge topology but requires one third less storage. The improved triangle-based data structure can easily be incorporated into VPF as a new primitive (Sec. 5.0). The introduction of this new primitive will, as shown in the next section, allow for the efficient storage and access of terrain elevation data as well as their integration with ground surface features.

Based on the preceding comparative analysis, DMAP recommends that the improved triangle-based data structure be utilized to encode TINs in EVPF. In the next section, the proposed data structure will be incorporated into VPF.

5.0 EVPF TABLE DEFINITIONS

Figure 5 shows the primitive directory of EVPF. A new optional primitive, TIN-face, has been added to topology level 3. This new primitive table comes with optional attribute, render-related attribute, and spatial index tables. The TIN-face table definition is shown in Table 11. In this table, vertices and adjacent triangles are stored in a counterclockwise fashion. Table 12 shows the definition of a modified connected node table. The new table includes an additional column, First-TIN, that implements the partial relationship between a connected node and its containing triangles.

While the TIN-face and the connected node table can be used to generate a wire mesh frame, a complete 3-D image requires more information about each primitive. To provide this information, a TIN-face attribute table must be introduced to store attribute data for each TIN primitive. Although stored at the primitive level, this table can be compared with an area feature table storing attribute data for the corresponding face primitives. The TIN-face attribute table stores attribute data for every triangle regardless of its relation or lack of relation to an area feature. The definition of the TIN-face attribute table is shown in Table 13.

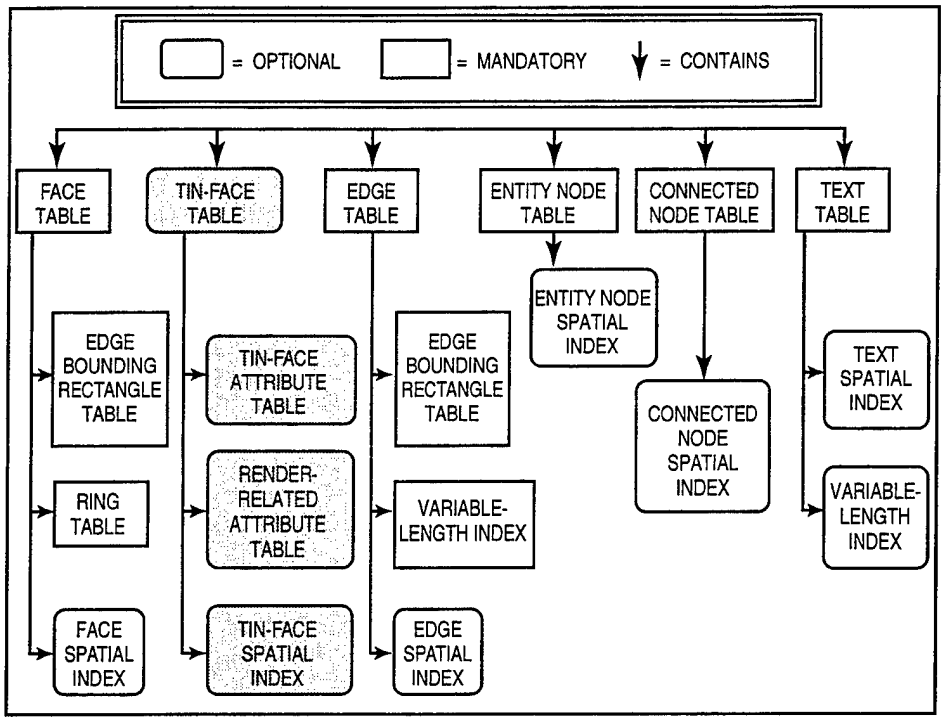

Fig. 5 — Extended primitive directory (level 3 topology)

Table 11 — TIN-Face Table Definition (filename: TIN)

(*Op/Man* = Optional/Mandatory Status, *M* = Mandatory, *P* = Primary Key, *N* = NonUnique Key, *I* = Long Integer, *K* = Triplet ID, "/" = "*or*")

COLUMN NAME	DESCRIPTION	COLUMN TYPE	KEY TYPE	OP/MAN
ID	TIN Primary Key	I	P	M
VERTEX1	First vertex (foreign key to the connected node table)	I	N	M
VERTEX2	Second vertex (foreign key to the connected node table)	I	N	M
VERTEX3	Third vertex (foreign key to the connected node table)	I	N	M
ADJACENT1	ID of triangle adjacent to current triangle along edge between Vertex one and two	I/K	N	M
ADJACENT2	ID of triangle adjacent to current triangle along edge between Vertex two and three	I/K	N	M
ADJACENT3	ID of triangle adjacent to current triangle along edge between Vertex three and one	I/K	N	M

NOTE: The vertices and adjacent triangles are stored in counterclockwise fashion.

Table 12 — Connected Node Table Definition (Filename: CND)

(same acronyms as Table 11, with *OF* = Optional Feature Pointer, *O* = Optional, *C/Z* = 2-coordinate structures, *B/Y* = 3-coordinate structures, *MI* = "Mandatory for TINs," and *M<n>* = "Mandatory: topology level <*n*>")

COLUMN NAME	DESCRIPTION	COLUMN TYPE	KEY TYPE	OP/MAN
ID	Node Primary Key	I	P	M
*.PFT_ID	Point Feature table ID	I	N	OF
CONTAINING_FACE	Always null (included for compatibility)	X	N	O
FIRST_TIN	TIN key (foreign key to the TIN-face table)	I	N	MI
FIRST_EDGE	Edge key (foreign key to the edge table)	I	N	M1-3
COORDINATES	Node Coordinates	C/Z/B/Y	N	M

Table 13 — TIN-Face Attribute Table Definition (Filename: TINATTR)

(same acronyms as Tables 11 and 12)

COLUMN NAME	DESCRIPTION	COLUMN TYPE	KEY TYPE	OP/MAN
ID	TIN Feature Primary Key	I	P	M
RENDER_VALUE	Render values for this triangle (foreign key to the RENDER.RAT table)	I	N	O
. . .	. . .	. . .	. . .	. . .
<Attribute n>	*n*th attribute	Any	Any	O
. . .	. . .	. . .	. . .	. . .

NOTE: The table TINATTR has a 1:1 relation with the table TIN.

To adequately render 3-D surfaces and objects, certain attributes must be defined for each triangle or for each vertex of each triangle. Among the attributes that could be defined are color, ambient reflection, diffuse reflection, specular reflection, emissions, and shininess. The color could be represented by a 3-tuple (R, G, B) of floating point values between 0 and 1. The RGB values represent the primary colors red, green, and blue. Black is represented by (0, 0, 0) and white by (1, 1, 1). The grayscales are represented in between by equal RGB values. The other values listed above are used to determine shading and the various effects light has on the object. To eliminate redundancy, these values can be stored in a related table and referenced by a number of individual triangles in the TIN-face table. The actual values needed by an individual rendering package can vary and as a result no standard fields are defined. The definition of the render-related attribute table is shown in Table 14.

Unlike existing features, it would be impossible to construct a TINed object from a single First-TIN as a face is constructed from a single start edge. For this reason, index and join tables would have to be extended and made mandatory for TIN-related features. A TIN feature index table

Table 14 — Render-Related Attribute Table
(Filename: RENDER.RAT)
(same acronyms as Tables 11, 12, and 13)

COLUMN NAME	DESCRIPTION	COLUMN TYPE	KEY TYPE	OP/MAN
ID	Render Primary Key	I	P	M
. . .	. . .	. . .	. . .	. . .
<Attribute n>	*n*th attribute	Any	Any	O
. . .	. . .	. . .	. . .	. . .

(TIN.FIT) can be used to define the relation between feature objects and the TIN primitives as well as to enhance the performance of primitive-to-feature and feature-to-primitive queries. In other feature index tables, there is at least one entry for each primitive and at least one entry for each relative feature. When more than one primitive is used to construct a feature, the feature has multiple entries in this table. The same is also true when a primitive corresponds to more than one feature. TIN and area features would generally be made up of one or more TIN primitives. To efficiently answer queries, these relationships must be defined explicitly. The TIN.FIT can be used to store these relationships. The existence of a specific TIN primitive in this index file demonstrates its use in constructing a TIN or area feature. The nonexistence of a specific TIN primitive implies that it is not used to construct any features.

A TIN feature index table can be employed to answer queries in both directions, whereas TIN feature join tables (*.IJT) can be used to enhance performance of feature-to-primitive queries. It would be used in the same manner as the area feature join tables. Each TIN and area feature table would have a join table with an entry for each TIN primitive used to construct a single feature.

6.0 TINS IN TILED COVERAGES

VPF allows cross-tile referencing via the introduction of a triplet ID data type. Each triplet ID begins with an 8-bit type byte depicting the format for the rest of the field. The three field names are ID representing the internal tile primitive ID, TILE_ID representing the external tile reference ID, and EXT_ID representing the external tile primitive ID. The first two bits of the type byte indicate the length of the internal ID; the second two bits indicate the length of the TILE_ID field; the third two bits indicate the length of the EXT_ID; the final two bits are reserved. The field length is either zero, one, two, or four bytes depending on the value stored in the two bits (i.e., 00 indicates zero bytes, while 11 indicates four bytes).

When a triplet ID is encountered, the type byte is analyzed and the resulting values used to find the correct primitive(s). In reference to an adjacent triangle, the existence (nonzero length) of the TILE_ID and EXT_ID fields indicates that the edge of the triangle is on a tile boundary and the adjacent triangle primitive is included in the tile specified by the TILE_ID. The EXT_ID is then used within the correct tile to find the appropriate primitive. This primitive should have corresponding triplet ID pointers to the original triangle.

In a tiled coverage, triangles will not be allowed to cross tile boundaries. This can easily be achieved by TINing each tile separately. In addition, care must be taken so that triangles at the edge

of a tile have at most three neighboring triangles (including the neighboring triangle in the adjacent tile). This can be achieved by using a constrained triangulation algorithm. In a constrained triangulation algorithm, predefined edges are used during the triangulation process. This ensures that the line and area features (surrounded by those edges) are preserved and exhibited by the resulting triangulation. This leads to a second method of generating TINs for tiled coverages. The tile boundaries would be defined as constrained edges. This would ensure that no triangles are bisected by the tile boundary while preserving the adjacency requirement.

7.0 A COMPREHENSIVE EXAMPLE

The example in Fig. 6 displays EVPF's ability to store elevation data in the form of a TIN. (Note: Primitive numbering is local to each tile.) Attribute data can then be assigned to each primitive by the introduction of a TIN feature table. This information transcends mere elevation points but provides actual details about the characteristics of the surface.

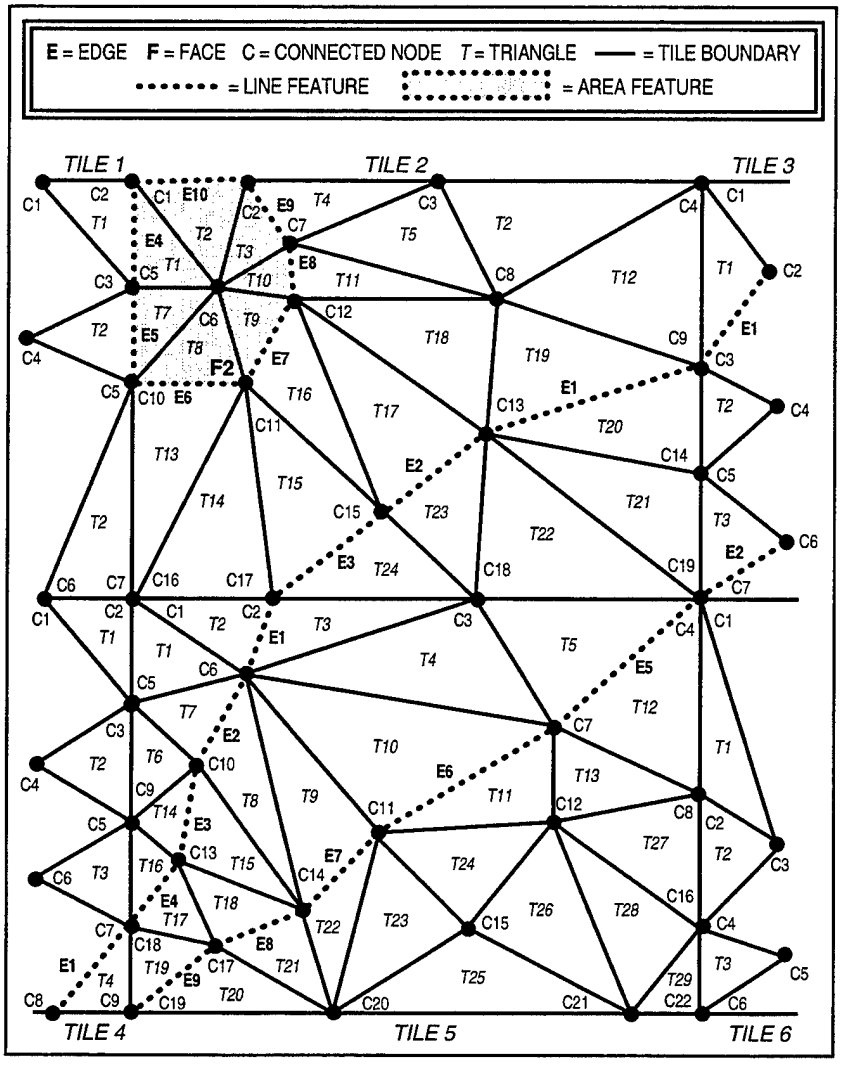

Fig. 6 — A sample tiled TIN

The example contains a tiled set of points that are used to construct a TIN in the presence of both area and line features. Connected node 6 of tile 2 is a local maximum point (a peak) where all adjacent nodes have a lower elevation. An arbitrary face can be used to represent this peak surrounded by edges 4 through 10 in tile 2. A face table would have a pointer to one of the edges from which the remaining bounding edges can be determined.

There are two line features contained in the example. The first is composed of edge 1 in tile 3, edges 1, 2 and 3 in tile 2, edges 1, 2, 3, and 4 in tile 5 and edge 1 in tile 4. This feature could be viewed as a river or a valley. The points along the line have a lower elevation in respect to any adjacent nodes not on the line. The second is composed of edge 2 in tile 3 and edges 5, 6, 7, 8, and edge 9 in tile 6 and can be viewed as a ridge line where all adjacent nodes not on the ridge have a lower elevation.

Tables 15 through 26 represent the partial encoding of the tiled sample TIN of Fig. 6.

Table 15 — Connected Node Table (Tile 2)

ID	FIRST_TIN	FIRST_EDGE	COORDINATES
. . .	. . .	. . .	. . .
5	1	Null	x_5y_550
6	1	Null	x_6y_6100
. . .	. . .	. . .	. . .
9	19	1	x_9y_930
. . .	. . .	. . .	. . .
13	17	2	$x_{13}y_{13}25$
. . .	. . .	. . .	. . .
16	13	Null	$x_{16}y_{16}30$
17	15	3	$x_{17}y_{17}20$
. . .	. . .	. . .	. . .

Table 16 — Connected Node Table (Tile 5)

ID	FIRST_TIN	FIRST_EDGE	COORDINATES
1	1	Null	x_1y_130
2	2	1	x_2y_220
. . .	. . .	. . .	. . .

Table 17 — TIN-Face Primitive Table (Tile 2)

ID	VERTEX 1	VERTEX 2	VERTEX 3	ADJACENT 1	ADJACENT 2	ADJACENT 3
1	1	5	6	(-,1,1)	(7,-,-)	(2,-,-)
2	1	6	2	(1,-,-)	(3,-,-)	(-,-,-)
. . .	. . .	. . .	. . .	. . .	. . .	. . .
14	11	16	17	(13,-,-)	(-,5,2)	(15,-,-)

Table 18 — TIN-Face Primitive Table (Tile 5)

ID	VERTEX 1	VERTEX 2	VERTEX 3	ADJACENT 1	ADJACENT 2	ADJACENT 3
1	1	5	6	(-,4,1)	(7,-,-)	(2,-,-)
2	2	1	6	(-,2,14)	(1,-,-)	(3,-,-)
. . .	. . .	. . .	. . .	. . .	. . .	. . .

Note: The adjacent triangle faces are of type triplet ID (Internal Primitive ID, Tile ID, External Primitive ID).

Table 19 — Edge Table (Tile 2)

ID	START_ NODE	END_ NODE	LEFT_ EDGE	RIGHT_ EDGE	LEFT_ FACE	RIGHT_ FACE	COORDINATES
1	9	13	(-,3,1)	(2,-,-)	1	1	$x_{1,1}y_{1,1}z_{1,1}x_{1,2}y_{1,2}z_{1,2}$
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
4	1	5	(10,-,-)	(5,-,-)	2	1	$x_{4,1}y_{4,1}z_{4,1}x_{4,2}y_{4,2}z_{4,2}$
5	5	10	(4,-,-)	(6,-,-)	2	1	$x_{5,1}y_{5,1}z_{5,1}x_{5,2}y_{5,2}z_{5,2}$
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
10	2	1	(9,-,-)	(4,-,-)	2	1	$x_{10,1}y_{10,1}z_{10,1}x_{10,2}y_{10,2}z_{10,2}$
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .

Table 20 — Face Table (Tile 2)

ID	RING_POINTER
. . .	. . .
2	2

Table 21 — Ring Table (Tile 2)

ID	FACE	FIRST_EDGE
. . .	. . .	. . .
2	2	4

Table 22 — TIN-Face Attribute Table

ID	RENDER VALUE*	SOIL TYPE	SIDE ATTRIBUTE ONE	SIDE ATTRIBUTE TWO	SIDE ATTRIBUTE THREE
. . .	. . .	. . .	. . .	. . .	. . .
10	15	43	1	2	2
11	15	43	2	2	1
. . .	. . .	. . .	. . .	. . .	. . .
20	22	45	3	1	2
. . .	. . .	. . .	. . .	. . .	. . .
30	24	46	3	1	2
. . .	. . .	. . .	. . .	. . .	. . .

**NOTE: The Render Value is a foreign key to the render.rat table specified in Table 23. The soil type and side attribute fields and their corresponding meanings can be documented in a primitive level value description table. The side attribute values are as follows: 1 – water flows out this side, 2 – water flows along this side, 3 – water flows in this side.*

Table 23 — Render-Related Attribute Table

ID	RED	GREEN	BLUE
. . .	. . .	. . .	. . .
15	0.75	0.40	0.60
. . .	. . .	. . .	. . .
22	0.20	0.30	0.90
. . .	. . .	. . .	. . .
30	0.10	0.90	0.20

Table 24 — Mountain Area Feature Table (Feature Class ID = 4)

ID	NAME	TOTAL AREA	HIGHEST PEAK
. . .	. . .	. . .	. . .
2	Mt. Sparta	600	100
. . .	. . .	. . .	. . .

Table 25 — Feature Join Table – Area to TIN

ID	MOUNTAIN.AFT_ID	TILE_ID	TIN_ID
. . .	. . .	. . .	. . .
10	2	2	1
11	2	2	2
12	2	2	3
13	2	2	7
14	2	2	8
15	2	2	9
16	2	2	10
. . .	. . .	. . .	. . .

Table 26 — TIN Feature Index Table

ID	TILE_ID	PRIM_ID	FC_ID	FEATURE_ID
. . .	. . .	. . .	. . .	. . .
21	2	1	4	2
22	2	2	4	2
23	2	3	4	2
24	2	7	4	2
25	2	8	4	2
26	2	9	4	2
27	2	10	4	2
. . .	. . .	. . .	. . .	. . .

8.0 PRIMITIVE ATTRIBUTE TABLES AND MULTIVALUED ATTRIBUTES

A classic example of the usefulness of primitive attribute tables can be seen in the following. In displaying a TIN-shaded relief representation of a surface, viewing software normally requires attribution for triangles or vertices that should be transparent to the user (e.g., color). This information can be stored in a primitive attribute table and render-related attribute table.

While not an extension to VPF per se, variable length fields could allow for attributes to have more than one value. Since not all attribute values describe mutually exclusive cases, defining multiple values to one attribute is a natural extension, particularly when knowing all of a particular feature's attribution is a necessity. An example is the AFA (Available Facilities Attribute), where AFA 9 implies the existence of a fueling station and AFA 23 implies the existence of a boat hoist. Clearly, a feature may have both capabilities, in which case the integers 9 and 23 would be stored in a variable length field of the attribute table.

9.0 SOFTWARE CONSIDERATIONS

To accomplish a full implementation of the EVPF, software must be constructed, not only to build tables (both VPF and the extensions) but also to compute the TIN data for storage in the TIN tables. DMAP has engineered a set of tools to accommodate the creation of an EVPF prototype, tentatively named Modeling and Simulation Extended Vector Product. However, the creation of an appropriate TIN from a given data set requires investigation. Examples of algorithms are available in the commercial geographic information system packages such as ARC/INFO (commercial geographic information system). Regardless of the algorithm, the data structures created in this report will be able to accommodate any given TIN.

10.0 CONCLUSIONS

The proposed EVPF permits the efficient storage and retrieval of TIN-based elevation data and the integration of such data with ground surface features. Based on the overall performance of four relevant data structures, an improved triangle-based data structure ideally suits the EVPF. An optional new primitive, TIN-face, has been added at topology level 3. To implement a partial relationship between a connected node and its containing triangles, a new column has been added to the connected node table. This new column points to one of the triangles containing the specified connected node. A TIN-face attribute table and a render-related attribute table have been added so that rendering and attribute data can be associated with each triangle, its edges, and its three extreme vertices. The render-related attribute table, which is essentially a primitive attribute table, would present a new concept: storing so-called "attribute information" (transparent to the user) at the primitive level.

Currently, VPF employs area and line feature tables to crudely represent real world 3-D surfaces and objects. The introduction of the TIN-face primitive allows for an elegant method of rendering terrain in three-dimensions. The problem of representing 3-D objects, such as buildings and bridges, remains as a needed future extension to VPF. Although the TIN face and TIN-face attribute tables can be used to store the primitives and their appearance for both terrain and objects, a TIN feature table (e.g., *.IFT) would have to be introduced at the coverage level to define the meaning or nature of objects. This table would function in a similar to the area feature table. A 3-D object would be triangulated and the resultant triangles stored in the TIN-face table. Each object would have a

single entry in the feature table. Intuitively, this seems to be a straightforward extension but presents additional problems. While the current design anticipates that extension, future and further consideration would have to be made for the development of the TIN feature index, TIN feature join and minimum bounding cube tables.

Finally, VPF allows for variable-length fields in its tables. Since the M&S community needs sufficient attribution, these variable-length fields should be utilized to the fullest. The capabilities of EVPF, and hence the M&S community, would benefit from the use of these fields.

11.0 RECOMMENDATIONS

DMAP's recommendations for a VPF more ideally suited for the M&S community may be summarized as follows:

- Incorporate the new primitive structure to allow for TINs encoded in the improved triangle-based data structure,
- Allow for additional attribute information stored at the primitive level (e.g., RENDER.RAT), and
- Use variable-length fields to store common multivalued attribute information.

The proposed georelational format, EVPF, is feasible and would be an essential step toward satisfying essential M&S requirements with vector data, given that one utilizes the relational database model. DMAP is currently developing an object-oriented database approach that will offer even greater improvements over VPF.

12.0 ACKNOWLEDGMENTS

This effort was sponsored by the Defense Mapping Agency's Terrain Modeling Program Office and the Defense Modeling and Simulation Office, under Program Element 630603832D, with Mr. Jerry Lenczowski as program manager.

13.0 REFERENCES

Baumgart, B. G., "Winged-Edge Polyhedron Representation," Report CS-320, Computer Science Department, Stanford University, Oct. 1972.

Department of Defense, "Vector Product Format – Military Standard," MIL-STD-2407, Department of Defense, May 1993.

Floriani, L. D., "Surface Representations Based on Triangular Grids," *The Visual Computer* **3**(27), 27–50 (1987).

Floriani, L. D., "A Pyramidal Data Structure for Triangle-Based Surface Description," *IEEE Computer*, Mar. 1989, pp. 67–78.

Jones, C., D. B. Kidner, and J. M Ware, "The Implicit Triangulated Irregular Network and Multiscale Spatial Databases," *The Computer Journal* **37**(1), 43–56 (1994).

Littlefield, K. E., "The Defense Mapping Agency's Digital Production System (DPS)," *Cartography and Geographic Information Systems* **22**(2), 119–127 (1995).

Polis, M. F. and J. McKeown, "Iterative TIN Generation from Digital Elevation Models," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1992, pp. 787–790.

Polis, M. F., J. G. Gifford, and J. McKeown, "Automating the Construction of Large-Scale Virtual Worlds," *IEEE Computer* **28**(7), 58–64 (1995).

Shamos, M. I., "Computational Geometry," Ph.D. Thesis, Yale University, New Haven, CT, 1978.

Shaw, K., V. Miller, B. Ray, R. Broome, T. Fetterer, M. Abdelguerfi, E. Cooper, and C. Wynne "An Extended Vector Product Format Profile for Modeling and Simulation," NRL/MR/7441--95-7704, Naval Research Laboratory, Stennis Space Center, MS, Apr. 1996.

Tachikawa, Y., M. Shiiba, and T. Takasao, "Development of a Basin Geomorphic Information System Using a TIN-DEM Data Structure," *Water Resources* **30**(1), 9–17, Paper No. 93062 (1994).

Woo, T. C., "A Combinatorial Analysis of Boundary Data Structure Schemata," *IEEE Computer Graphics and Applications* **5**(3), 19–27 (1985).

Appendix

ACRONYM LIST

3-D	three dimensional
AFA	Available Facilities Attribute
ARC/INFO	commercial geographic information system
DMA	Defense Mapping Agency
DMAP	Digital Mapping, Charting, and Geodesy Analysis Program
DTED	Digital Terrain Elevation Data
EVPF	Extended Vector Product Format
M&S	Modeling and Simulation
RGB	Red Green Blue
TIN	Triangulated Irregular Network
VPF	Vector Product Format